10/509030
27 SEP 2004
PCT/CN03/00222

证　明

REC'D 28 MAY 2003
WIPO PCT

本证明之附件是向本局提交的下列专利申请副本

申　请　日：2002 03 28

申　请　号：02 1 08163.8

申请类别：发明

发明创造名称：软件模拟序网N＋1个并行程序的结构和方法

申　请　人：徐肇昌

发明人或设计人：徐肇昌；吴东明

PRIORITY
DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

中华人民共和国
国家知识产权局局长 王景川

2003 年 4 月 15 日

02.03.26

权 利 要 求 书

PRE/CN02/2026/SGL

1. 一种基于单机环境的并程结构模块，其特征在于包括:

N+1 个并程分支程序模块（N≧1），通过 N+1 个独立结构的分支程序时分运行（突出和并行运行的时分区别）实现并程模块的运行，在每个所述的并程分支程序的读并程数据、写并程数据、和并程数据一致化三类序网指令的子程序支持下，完成并程分支程序的数据传送和同步；和

一个管理程序模块，用于响应来自所述并程分支程序的信息，支持并程分支程序的悬挂、就绪、和运行状态。

2. 根据权利要求 1 所述的并程结构模块，进而包括一个并程分支程序悬挂处理模块，用于处理当前并程分支程序的悬挂、就绪、和运行状态。

3. 根据权利要求 1 的并程结构模块，进而包括一个并程进入/退出模块，所述并程进入/退出模块响应所述时分管理程序模块的请求，对每个实际应用的并程进行初始化处理和对并程的退出进行处理。

4. 根据权利要求 1 所述的并程结构模块，其中， 第 N+1 个并程分支程序执行一个并程数据的序，用数据一致化操作表示。

5. 根据权利要求 4 的并程结构模块，其中，所述第 N+1 个分支程序中的令牌一致化操作和并程数据对应，直接通过所述第 N+1 个程序控制并程数据一个个地一致化，其中，并程数据一致化可以执行的条件是该并程数据已经被其他并程分支程序写过，否则该并程分支程序就要悬挂，一致化执行后，表示该并程数据被其他并程分支程序读取时呈有效，否则那些读并程分支程序就要进入悬挂。

6. 根据权利要求 5 的并程结构模块，其中，在并程嵌套和子并程调用时，所述第 N+1 个程序可以连接和控制子并程的调用许可和并程数据序的关系。

7. 根据权利要求 5 的并程结构模块，其中， 所述 N+1 个并程包括三种并程子程序，用于处理并程数据的读、写、和一致化操作，所述子程序具备令牌的设置、检测、和处理能力，并支持并程分支程序的同步和并程数据的传送。

8. 一种基于单机环境的 N+1 个并程分支程序(N≧1)的运行方法，其特征在于包括以下步骤:

初始化处理;

进行所述并程分支程序结束检查，检查所述并程分支程序的开关标志，其中:

如果全部并程分支程序为关，表示并程运行已经结束，则处理并程结束的记录以及和外部程序的连接方式;和

如果有一个并程分支程序是开，则进行分支程序的悬挂状态检查; 其中，

如果检查结果是全部并程分支程序处于悬挂状态，则表示尽管有并程没有执行完，但该并程分支程序已不能运行下去，从而查找并程分支程序不能运行的原因，并根据该原因处理并程分支程序的退出;

如果检查结果是有一个以上的并程分支程序处于就绪状态，则进入并程分支程序排队处理;

选择一个就绪的并程分支程序进行分支程序排队处理;

选定并程分支程序以便对其参量加载并恢复该并程分支程序先前悬挂时的状态；和

进入该选定的并程分支程序运行。

9. 根据权利要求 8 所述的并程运行方法，其中，所述初始化处理包括并程分支程序的参量的加载处理和并程分支程序的标志区的清 0 处理。

10. 根据权利要求 9 所述的并程运行方法，其中，所述并程分支程序的参量的加载处理包括：

打开并程分支程序的 N+1 分支程序标志 ON；

设置各并程分支程序的入口地址；

设置相应的寄存器数据初始值；和

将并程标志区和并程分支程序数据标志清 0。

11. 根据权利要求 8 所述的并程运行方法，其中所述并程分支程序包括一个写数据子程序，其包括以下步骤：

对一个并程数据进行写操作；

检查该数据“曾一致化无效”标志；如果该标志为有效，表示该数据曾有过数据一致化操作，但因为一致化标志无效，使该一致化操作实际上没有能执行，导致一致化并程分支程序进入悬挂状态；

在数据有效的情况下，将就绪标志中该一致化并程分支程序的状态位从悬挂改为就绪；

在数据无效的情况下，建立“一致化有效”标志，使允许一致化并程分支程序对该数据执行一致化的操作；和

写数据子程序返回。

12. 根据权利要求 8 所述的并程运行方法，其中所述并程分支程序包括一个写数据子程序，其包括以下步骤：

检查该数据的“一致化有效”标志；其中，

如果该数据的“一致化有效”标志无效，则悬挂该并程分支程序；

设置“曾一致化无效”标志；

进行当前一致化分支程序的悬挂处理，保存一致化并程分支程序的当前运行状态，以便在今后该并程分支程序返回时使用；和

退出该并程分支程序，转并程分支程序悬挂入口，以便重新选择新的并程分支程序；

如果该数据的“一致化有效”标志有效，则执行数据和令牌的一致化操作；

检查该数据的 N 位“曾读无效”标志；以及

数据一致化子程序的返回。

13. 根据权利要求 12 所述的并程运行方法，其中，在所述检查该数据的 N 位“曾读无效”标志步骤中，如果 “曾读无效”标志有效，则根据“曾读无效”标志的内容，使相关的并程分支程序状态从悬挂转入就绪。

14. 根据权利要求 8 所述的并程运行方法，其中所述并程分支程序包括一个读数据子程序，其包括以下处理步骤：

检查该数据的“读有效”标志，如果该数据的“读有效”标志有效，读数据；以及

读数据子程序的返回。

15. 根据权利要求 14 所述的并程运行方法，其中，在所述检查读数据的“读有效”标志步骤中，还包括以下步骤：

如果该数据的“读有效”标志无效，设置 N 位“曾读无效”，分别对应不同的并程分支程序；

对当前并程分支程序进行悬挂处理，保存并程分支程序的当前运行状态，以便在今后该并程分支程序返回时，可以从悬挂点切入继续运行；以及

退出该并程分支程序，转并程分支程序悬挂入口，重新选择新的并程分支程序。

说 明 书

PRE/CN02/2026/SGL

软件模拟序网 N+1 个并行程序的结构和方法

本发明所属技术领域

本发明涉及计算机技术，尤其涉及计算机软件的一种模块结构类型。

在先有技术中，对并行流程图的看法是“N 个流程和数据的互连”，先有发明序网计算机认为并行流程图实际有 N+1 个序，以及在 N+1 个序的结构下的分布数据令牌结构，如图 1A 所示，称为序网。在用计算机实现并行流程图时，先有发明认为 N+1 个序是基本要素，必须有 N+1 个流来对应，是（并行）计算机可以运行的基本特点。在 N+1 个流序的结构条件下，为了维持并行流程图的功能，先有技术采用了一种分布令牌结构，和第 N+1 个流序执行（数据）令牌一致化的结构。因此，先有发明生成一种具有 N+1 个流的计算机结构，该结构也是图 1A 的形式，并定义为序网计算机。

序网计算机结构包括 N+1 个并行程序和分布数据令牌的连接结构。由于从程序角度看，序网只是用 N+1 个并行程序（代替原来 N 个程序）的运行控制方法。因此，本文从运行控制角度，称序网结构为“并程”或“N+1 并程”。同时，将过去技术中，以 N 个并行程序的处理结构，称为“N 并程”。 尽管并程或 N 并程概念不同，其分支程序间要传送的数据是一样的，本发明将上述并程和 N 并程的互连数据称为并程数据，在下面描述也会简称为数据。

N+1 并程结构如图 1A 所示，并程包括 N+1 个程序 100、101、102、103（本图和本发明描述中取 N=3，以后不再作说明），和四个

分布（数据）令牌 108。新增加的第 N+1 个程序 100 又称为一致化程序。分布（数据）令牌 108 在程序中表现为三种新的指令，包括写数据指令 TSET 104，读数据指令 TTST 105，和并程数据一致化指令 TCOS 106。图中，箭头表示数据传送路径，标记 107 表示分布令牌的一致化操作。其运行原理如下：

- 当 S1、S2、或 S3 程序运行到写数据指令 TSET 104 时，表示写数据。在写数据的同时，设置令牌=1，表示该并程数据的一致化操作允许执行。
- 当一致化程序 100 运行到一致化指令 TCOS 106 时，执行一个并程数据的一致化操作。在执行该分布数据令牌的一致化操作 107 时，一致化操作是数据和令牌从源 104 出发，在该分布数据令牌 108 内的广播。在一致化传送中要测试令牌，若令牌无效，该一致化传送重新执行。直至出现有效令牌，才进入下一条指令。
- 当 S1、S2、或 S3 程序运行到读数据指令 TTST 时，表示读数据。在读数据的同时，进行令牌测试。若令牌无效，则程序要重新读数据，直至出现有效的令牌，才进入下一条指令。

显然，在多机结构下，N+1 并程依靠一个新增的一致化程序 100，在分布数据令牌结构的支持下，实现了并程内部的并行计算，包括并程分支程序的同步和数据传送。其中，一个数据的传送需要三条分布在不同程序中的指令，二次同步协调。它们是写数据（产生令牌），令牌一致化（对写令牌检测和协调），读数据（对一致化令牌检测和协调）。一致化程序 100 的特点是驱动并程数据的序运行，没有数值计算内容。

事实上，在多机结构下，N+1 并程依靠一个新增的一致化程序 100,另外还支持了并程的嵌套功能。在结构上，并程的嵌套是二个并程有父子关系，父并程的 N+1 个分支程序同时为子并程的 N+1 个分支程序所调用（中断）。在数值计算模型上，如果父子并程没有数据连接关系（如来自外部中断事件的子并程），嵌套时可以在父级 N+1 个分支程序的任何位置。如果父子并程有数据关系（程序结构的需要），即算法要求子（嵌套）并程和父级并程有上下文关系。举例来说，如果在 1 号分支程序中发出的一个子并程的嵌套调用，子并程计算需要父并程所有 N+1 个并程分支程序准备一定数据，表示父级各个分支程序各自要运行过某个程序点，才能允许子并程的调用，这是数据的上下文联系的需要。这实际上表示子并程的调用（嵌套）是串型的，表示子并程调用包括二个参量：在父并程分支程序中的子并程调用指令（编程很方便），和父并程运行顺序相关的调用许可控制。

上述 N+1 个程序点定义了子并程调用位置的许可区。用 N 个分支程序控制这个上下文关系比起用一致化程序 100 来控制要难一个量级。控制包括确立许可位置，和并程数据序的关系，以及实际的执行，其中，并程数据序就是上下文的数据。实际的许可执行方法很多，可以用一个并程数据加上特定的解释来代替子并程调用许可信号，也可以专门设置新的指令。

因此，支持子并程嵌套调用的关键是父并程有一个并程数据序的实际执行过程，和一个程序结构来控制并程数据序的运行。有了上述二个特征，子并程的许可可以在程序控制下灵活处理，并且可以编程定义和上下文数据的有序关系。并程嵌套能力对于程序的通用性结构能力至关重要，显然，并程的第 N+1 个程序是支持并程嵌套的关键条件。

序网计算机的并程模块有下列特征:

- 序网计算机的并程模块结构简洁
- 并行模块具备一个数据序特征，支持并程的并行计算，数据序来自第 N+1 个程序 Sc。
- 多个并程模块之间允许结构上的嵌套。
- 各分支程序是独立的程序模块结构。

进一步，可以创建单机调用并程的指令，实现并程模块之间的嵌套结构。创建在第 N+1 程序中的子并程调用许可功能，实现子并程的调用许可。

- 创建单机中的调用并程指令，实现并程调用。
- 一条调用指令有 1 到 N 个并行的并发调用能力。
- 利用现有的并程数据结构来代[illegible]调用许可。

序网计算机技术明确了并行流程图有 N+1 个并程分支程序，但是，目前大多数人对并行流程图的概念还是 N 个流程。这个观念的差异，在多机结构下，限制了并行技术的发展；在单机环境下，这种差异变小，不易分辨。但是，在一些特定的情况下，这种概念差异就会在具体结构中表现出来。

本发明是从序网计算机的 N+1 并程分支程序的角度来看并程进入下列单机环境的二种结构。

根据并行计算的流序匹配原理，并行模块的本质是有 N+1 个序，计算机需要 N+1 个硬件的流支持，实现 N+1 个流序匹配。在单机环境下，计算机只有一个流，因此，必须先把 N+1 个序合并成一个序，才能实现流序匹配。合并的核心是对 N+1 个序进行时分处理。同时，

尽管过去在理解上是做 N 个序的合并，实质上做的还是 N+1 个序的合并，存在一种实际操作和概念理解上的差异。

先有技术有下列二种合并 N+1 个序为一个序的方法，形成不同的结构。图 1B 所示的是一种直接组合的简单结构，没有并程分支程序的独立结构问题，使上述实际操作和概念理解上的差异被掩盖了。由于图 1C 要求独立的分支程序结构，上述差异的影响充分表现出来，形成一种 N 并程结构，即用 N 并程结构代替 N+1 并程的实际结构。

图 1B 是直接合并 N+1 个序成为一个序的结构图。这是将 N+1 个分支程序合并成一个不可分割的串型程序的方法。其运算结果不变，但运行过程不同，因此，是一个计算过程范畴的问题。

如图 1B 所示，[illegible]2 112，S3 113 是三个程序，各自都已经拆散成多个程序段 [illegible]125。各段右侧的圈表示该程序段产生的并程数据（由写数据指令 104 产生），箭头线指向该并程数据在其他程序段需要应用（读数据指令 105）的位置。然后，将这些拆散的段组合，组合的必要条件是组合后的序和全部 N+1 个序匹配（因此，拆散时的分段方法也是有要求的）。显然，所有的箭头线必须是向下，而不能向上。该特点，就是并程数据一致化序 Sc 110 中的序。合并后成为一个程序 126 是一个串型的程序，是一个不可分割的模块。

N 并程概念是没有第 N+1 个序 Sc 110，由于 Sc 和数值计算结果没有关系，加上在单机环境下，不再需要一致化数据的传送，只要合并后的单机程序 126 满足全部 N+1 个序的关系即可，因此，对 N 并程或 N+1 并程二个概念来讲，上述的合并实现数值计算的结果可以是一样的。这是造成 N+1 概念长期没有得到重视的原因之一。

在单机环境下，本图所示的合并方式是最常用的方法。通俗的讲，第 N+1 个流序就是数据先写后读的排序模式。经过合并处理，一个并程模块变成一个单机中常见的（子）程序模块。

如果一个经过合并处理的并程模块是一个中断程序或子程序，正在运行的程序也是一个经过合并处理的并程模块，就相当于实现了并程模块的嵌套。一个子并程调用在（多机的）序网计算机环境中，有分布的调用指令和串型的调用许可二个条件，以便适应上下文的算法需要；但是，在单机环境的图 1B 结构下，N+1 个序合并成一个序，调用指令的位置隐含了调用许可，以满足上下文的算法需要。当然，所有描述只是众所周知的单机中断程序或调用子程序在本发明视角下的一种解释。

N+1 个序直接合并成一个序的并程模块结构非常简单，可以支持并行计算（同步等运行），可以支持并程间的可嵌套性能，都是图 1B 方法的优势。但是，第 N+1 个序 Sc 中的并程数据序必须有固定的序结构，子并程的调用位置也必须固定是不足之处，使上述直接合并方法比较适合于计算，而不适合多程序的管理。各并程分支程序没有独立的模块结构也是一个不足，不能实现（某些特定场合下）各分支程序加载或卸载等功能。

综上所述，该结构有下列特点：

- 并程模块结构简单。
- 并程模块支持并行计算，但计算功能受并程数据序固定化的约束。
- 并程模块允许并程间嵌套。
- 并程模块中各并程分支程序的程序结构不是独立模块。

在某些实际应用中需要并程各分支程序在机器码层保持独立的模块结构（例如需要对分支程序加载和卸载操作），先有技术基于并行流程图有 N 个流程的概念，产生如图 1C 所示的 N 并程结构。N 并程的 N 个分支程序有独立结构。在单机环境下，一个简单的设想是让 N 个分支程序时分运行，时分需要一个 N 并程管理模块，形成如图 1C 所示的，包括一个 N 并程管理模块 130 和 N 个独立结构的 N 并程分支程序 131、132、133 的结构。要保持 N 个独立并程分支程序和传统程序概念一致，N 并程管理模块 130 不能是一个简单的时分问题，需要解决下列问题:

- N 并程分支程序的时分管理。

这是一个众所周知的算法模型。N 并程管理模块 130 是父级结构，N 个并程分支程序是子级。模块 130 进行时分调度，选定一个并程分支程序运行。该分支程序在运行中，如果条件不能满足，不能运行时，分支程序自动进入悬挂，并退回到模块 130，使可以选下一个并程分支程序。如此往返，一直到全部并程分支程序执行完毕，并有 N 并程管理模块 130 运行结束，退出。因此，单纯的时分管理结构并不复杂。

如果 N 个程序是并行计算程序，相互的数据交换使必须有一个正确的并程数据序存在，上述时分算法结构可以使 N 并程全部执行完毕，但实际执行的并程数据序，在时分模块 130 的处理下是不加控制的，即实际的并程数据序是不确定的。不能确定的并程数据序，使实现 N 并程相互间的嵌套需要的上下文控制变得十分困难。同时，由于没有第 N+1 个程序，使分布在各分支程序中的调用 N 并程的指令也不能通过许可来排序，影响了 N 并程之间的嵌套能力。

- N 并程分支程序的运行同步。

根据本发明的观点，同步过程的本质是执行一个正确的并程数据序：

N 并程的结构没有第 N+1 个一致化程序和指令，因此，一个数据的传送从“写” “一致化”和“读”三步的一般状况，转为二步结构。如：“写_一致化”和“读”。写_一致化在各个分支程序中启动，在客观上，各个分布的写_一致化操作仍必须有一个正确的并程数据序关系。在读数据发现检测无效时，该分支程序要退出，返回到父级的 N 并程时分管理模块 130。通过时分管理模块的调度，可以使全部的分支程序得到执行，完成 N 并程的运行。其中，要求各分支程序之间的数据序在逻辑上是一致的，不能出错。

- N 并程分支程序间的数据通讯。

并程分支程序间的通讯有很多方法，在采[illegible]结构的情况下，通讯包的结构和解释也进入了模块 130 [illegible]，他们和程序同步管理是不同的处理。当然，数据通讯也可以脱离于 N 并程管理模块 130 的结构范围，独立处理。

N 并程在先有技术中是软件设计，有些结构并不公开，因此本发明的评论不能完全。但是，从根本上讲，N 并程运行缺少并程数据序和独立的第 N+1 个程序，影响了 N 并程的可嵌套性和并行计算能力。另外，实际上 N 并程运行时，其内部也必须有类似于处理第 N+1 个流序的内容，并通过其他结构来表现。例如，消息队列是一种进程间的通讯方式，是一种互连信息的排队，但它不是程序，没有程序那样主动运行的驱动能力。

N 并程的情况综述如下：

- N 并程管理模块处理功能太多，结构复杂。

- N 并程支持并行计算，但由于其分支程序连接是基于通讯，有粒度和并行功能的限制。
- N 并程之间的嵌套关系有困难。
- N 并程分支程序有独立结构，支持分支程序的加载和卸载。

N 并程和图 1A 所示的序网功能相比，在并行性能上还不如多机系统。情况如下：

- 单机环境和多机环境差别。
- N 并程结构相当复杂。
- N 并程运行中没有并程数据序，没有并程数据序处理同步和计算的能力。
- N 并程没有第 N+1 个程序可以支持“子 N 并程”的调用许可，不能和并[illegible]合，难以实现嵌套需要的上下文处理。
- N 并程有独立的分支程序结构。

本发明的目的

先有序网计算机技术表明，N+1 并程是并行流程图的自然特征。在该自然结构下，并程有 N+1 个在同一个层次上运行的分支程序，由 N+1 个并程分支程序控制和实现同步和数据传送，并支持并程的嵌套。本发明进入单机环境，要求能保持上述 N+1 并程分支程序独立运行、以及运行中的同步和数据处理，支持并程的嵌套结构的特点。如果可以保持这些性能，在结构上只要增加一个 N+1 个并程分支程序的时分管理模块即可。单纯的时分管理比较容易处理，只需要将序网计算机的 N+1 个分支程序的等待状态，转化为悬挂、就绪等类似于进程的状态结构。时分管理模块和 N+1 个并程分支程序肯定是不同层的。和先有技术图 1C 所示的 N 并程结构相比，本发明

的第 N+1 个并程分支程序是一个主动运行的、执行并程数据序，和支持 N+1 并程嵌套的程序。显然，符合其结构的自然本性。从而使本发明的 N+1 并程结构成为一种一般性结构

并程是一种符合模型自然特征的结构，在多机系统中，序网计算机是一种符合这个自然规律的计算机模型。在单机环境中，也应当建立符合这个自然规律的特征。

因此，本发明的目的是创建一种单机环境下的具备 N+1 个运行程序的并程结构。其中，第 N+1 个并程分支程序是一个执行并程数据序的程序和支持并程嵌套结构的程序。

根据本发明的目的创建的一种单机环境下的并程结构包括:

- 简单的处理 N+1 个并程分支程序的并程时分管理结构;
- N+1 个并程分支程序建立并程数据序，实现并行计算中的运行同步和数据传送功能;
- 支持并程和并程间的嵌套关系; 以及
- 并程 N+1 个分支程序是独立的模块化结构。

上述四项功能，不仅在单项上集中了图 1A，图 1B 和图 1C 的优点，更有综合后的创新优势。

本发明的目的是建立一种将并行功能合理分配的结构，在完成上述并行性能后，还进一步创造出新的计算资源。包括:

- 由第 N+1 个程序控制的“并程数据序”，和子并程调用许可，表现并程运行的过程特征。
- 有余力的第 N+1 个程序资源。
- 有余力的并程时分管理模块。

附图说明

本发明自此开始进入本发明的结构描述。本发明的新颖独特的特征、使用方法、发明目的等内容在参照以下将要详细描述的实施例和相关的附图后，会得到很好的理解。其中：

图 1A 是先有技术序网计算机的结构和功能图。

图 1B 是先有技术直接合并 N+1 个序成为一个序的结构图。

图 1C 是先有技术 N 并程结构图。

图 2A 是本发明的 N+1 并程结构图。

图 2B 是本发明的 N+1 并程的实用化结构图。

图 2C 是本发明的并程标志图。

图 2D 是本发明的一个并程数据标志图。

图 3A 是本发明并程管理程序的流程图。

图 3B 是本发明的写数据子程序流程图。

图 3C 是本发明的数据一致化子程序流程图。

图 3D 是本发明的读数据子程序流程图。

图 3E 是本发明的 N+1 并程分支程序的结构图。

下面，结合附图，我们将进一步清楚地说明本发明的各种目的和优点。

具体实施方式

图 2A 是本发明的 N+1 并程结构图示出了在单机环境下，用软件来实现先有技术的“一种基于分布结构的并行模块结构及其并行处理方法”的实施例。这也是在单机中实现并程的一般化结构。

本发明的结构包括时分管理模块 201，N+1 个并程分支程序模块 207，并程分支程序悬挂处理 202。参照图 1A，本发明在单机环境下实现并程的方法是：保持 N+1 个独立的并程分支程序结构 207，保留图 1A 具备的自动的并程分支程序间的运行同步和数据传送能力，在这个基础上，再进行分支程序的时分管理。时分管理就是利用并程分支程序的悬挂、就绪、和运行状态，控制分支程序的分时运行。

每一次，N+1 个并程分支程序模块 207 中只有一个分支程序可以运行。该分支程序中原有的读并程数据、写并程数据和并程数据一致化三类序网指令，在本发明中用三个软件子程序来代替。其中，写并程数据子程序能直接执行，而读并程数据、并程数据一致化子程序包含可执行条件的检查。如果数据无效，则检查时将发现分支程序不可继续运行。模块 207 中的该分支程序停[illegible]模块 202，对该分支程序进行悬挂处理。如果该并程分支程[illegible]束，则通过 208 转模块 201。

模块 207 中没有运行的并程分支程序处于“运行 ON”或“关闭 OFF”状态，运行 ON 状态又分悬挂、就绪二种情况。每个分支程序都有相应的数据、程序地址等专用存储区。这些概念在众所周知的进程或线程管理中很普遍。其中，第 N+1 个分支程序 203 尽管执行的是并程数据的序，是新的内容，但在悬挂、就绪等处理方式上是一样的。

分支程序悬挂处理模块 202 接受来自并程分支程序模块 207 的分支程序不能运行的信息，包括读无效和一致化无效二种情况。一致化无效是因为该并程数据没有被写过，读无效是因为该并程数据没有被一致化操作过。二种情况都在相关子程序检查并程数据标志

时发生，表示该分支程序不能继续运行。分支程序悬挂处理模块 202 处理当前并程分支程序的悬挂，现场保护等内容，然后转模块 201。

时分模块 201 的目标是选定一个新的并程分支程序。时分模块具备全部并程分支程序的状态信息，主要是各分支程序的悬挂，就绪和开关状态。这些信息不断地在修改，例如写并程数据子程序需要检查该数据位置是否曾发生过一致化无效，使一致化程序 PPc 203 进入了悬挂状态；如果是曾有悬挂处理，现在需要修改相关并程分支程序的悬挂状态为就绪状态。时分模块按照一定的优先权规则选出模块 207 中下一个处理就绪状态的并程分支程序，然后，进行该分支程序的恢复处理，并进入该分支程序运行。时分管理模块不直接处理并程分支程序的同步或数据通讯。模块 207 选出下一个并程分支程序的方法，根据需求的不同，模块 207 可以有各种不同的优先权规则设计方案。例如，在[illegible]排队来建立优先权时，先启动的分支程序可以优先执行。

当全部分支程序运行结束后，时分管理模块 201 才停止选择。当全部并程分支程序处于悬挂状态时，表示整个并程不能运行，时分管理模块 201 也停止选择，执行退回到上一级并程或等待输入数据等操作。

本发明和图 1C 的结构的核心区别在于，并程有第 N+1 个独立运行的并程分支程序 203，和 N+1 个程序的运行特征，N+1 个并程分支程序有同等地位。其中，第 N+1 个并程分支程序执行一个并程数据的序，用（数据）一致化操作表示。它突出了并程的数据序特征和独立的结构，和其他 N 个分支程序处理的计算内容完全不同。

在先有技术中，第 N+1 个程序是含有“数据和令牌一致化”指令的程序，完成多机之间的数据一致化和令牌一致化（传送和判别）操作。在单机软件环境下，该指令转为一致化子程序，令牌也改为

并程数据标志，用存储器来代替，令牌的判别也改为软件来判别标志内容。同时，在单机环境下，数据和令牌共同传送的情况也变了，并程数据的传送（一致化）操作已经不需要了，但是，令牌一致化的处理仍旧需要。在软件方式下，令牌一致化就是对并程数据标志位的操作。

第 N+1 个分支程序中的令牌一致化操作是和并程数据对应的，直接通过第 N+1 个程序控制并程数据一个个地一致化。并程数据一致化可以执行的条件是该并程数据已经被其他并程分支程序写过，否则该并程分支程序就要悬挂；一致化执行后，则表示该并程数据被其他并程分支程序读取时呈有效，否则那些读并程分支程序就要进入悬挂。

第 N+1 个分支程序是一个程序，对一个程序资源来讲，完成一致化操作并程数据序后，仍有很多程序的资源有待利用。当然这些资源的利用已经超越了一个纯粹的并行流程图计算的需要。在并程嵌套和子并程调用方面，第 N+1 个程序可以连接和控制子并程的调用许可和并程数据序的关系，在此，并程数据序可以为子并程调用提供算法需要的数据的上下文关系。本发明第 N+1 个程序有主动运行的驱动能力，因此，可以通过 N+1 个程序直接实现分支程序的同步和并程数据状态的管理，并支持并程嵌套。

在单机中，并程的三个序网指令是用三个子程序来代替的，子程序在程序中代替原来的读、写、和一致化指令。因此，N+1 个并程 207 内部包含了三种特有的并程子程序，处理并程数据的读、写和一致化操作。子程序具备令牌的设置、检测、和处理能力，支持并程分支程序的同步和并程数据的传送。在软件模拟情况下，每一个并程数据需要有相关的并程数据标志。

独立的并程分支程序模块结构是本发明的目标之一，无论在在程序的编程阶段，程序的机器码阶段，它们都是独立的模块结构。

本发明的时分管理模块 201 可以和 N+1 个分支程序 207 一起封装，成为并程模块 200 的一部分，因此，并程在运行中无需操作系统的管理。由于并程内部具备了时分、同步、数据互连管理，因此，可以组成并程间的多级嵌套结构。

本发明并程的用户编程界面是 N+1 个并程分支程序 207。其中 N 个程序是常规的单机程序，在 N 个程序中，通过对并程数据的读和写来实现程序之间的数据互连，和共享存储器的读写概念一致。在编译阶段，编译软件将共享存储器的读或写转为调用一个（读/写）子程序形式，并建立和并程数据相关的并程数据标志存储区。第 N+1 个程序是上述并程数据的一致化操作，没有数值计算的内容。一个一致化操作包含一个并程数据，用户编程只需要顺序列写并程数据或含并程数据的一致化指令。和图 1C 的 N 并程方式相比，并程数据的序是一种新的输入信息类型，并通过程序来驱动这个序的执行。

模块的运行过程如下：时分管理模块 201 的功能是定义一个并程分支程序进入运行。全部 N+1 个并程分支程序中，第 N+1 个并程分支程序 203 和其他 N 个并程分支程序执行的功能是不同的。

模块 201 定义 N 个并程分支程序 204、205、206 的运行情况如下：并程分支程序进入程序运行后，如果条件满足，可以一直运行到结束，在分支程序结束后返回时分管理模块 201。时分管理模块 201 再决策，进入下一个就绪的并程分支程序。

如果并程分支程序运行遇到读或写数据，处理如下：

- 在读数据时，如果该并程数据有效，则读数据完成，进入下一条指令。如果发现该并程数据无效（这个数据要来自另外一个并程分支程序），分支程序将不能继续运行，读数据的子程序自动在该数据的数据标志中设置“曾读无效”标志，转模块 202。模块 202 中保护该并程分支程序现场，使退出该分支程序，再转时分管理模块 201，重新选择新的处于就绪状态的并程分支程序。

- 在写数据后，要设置数据的“一致化有效”标志。同时，还需要检查该数据的“曾一致化无效”标志（表示第 N+1 个并程分支程序 203 已经处理过此数据而没有成功，并进入了悬挂状态）。如果有标志，写数据的子程序则将一致化分支程序的状态从悬挂改为就绪，然后返回。该并程分支程序继续运行。写数据子程序不能使并程分支程序进入悬挂。

模块 201 定义第 N+1 个并程分支程序 203 的运行情况如下：在执行某个并程数据的一致化前，需要对该数据的“一致化有效”标志进行测试。

- 如果标志无效，一致化子程序设置“曾一致化无效”标志，然后转模块 202。模块 202 保护并程分支程序 203 现场，使退出该分支程序。再转时分管理模块 201，重新选择新的处于就绪状态的并程分支程序。

- 如果标志有效，转入数据一致化的执行。单机环境下没有数据的传送，一致化操作的实质是处理标志位。包括设置“读有效”的标志；检查所有 N 个并程分支程序在该数据的“曾读无效”标志，并根据检查结果，使有关并程分支程序的状态从悬挂转为就绪，然后返回。一致化分支程序继续运行。

当时分管理模块201完成了全部N+1个并程分支程序203、204、205、206的运行之后，表示并程运行结束。

因此，总结本发明的并程特征如下：在单机环境下，一个并程将包括N+1个并程分支程序模块，和一个时分管理程序模块201。另外，还有三个“写，读，和一致化”子程序。

- 根据并程的同步处理规则，读数据无效等待需要在单机中用软件仿造实现。
- 根据并程的数据令牌传送规则，一致化无效也需要在单机中用软件仿造实现。

并程分支程序管理模块201的功能只需要包括因单机环境而产生的时分管理。

- 利用并程的同步处理建[illegible]程分支程序的等待和运行状态。
- 时分管理将并程分支程序的等待转为就绪，悬挂，和运行的管理。

上述关于本发明的结构描述，实际上已经包括了支持并程嵌套的二个基本结构条件：

- 并程有第N+1个独立的程序
- 第N+1个程序中管理并程数据序运行

子并程调用需要实现和父级并程中N+1个分支程序的上下文连接。包括下列二个可调用条件，

- 在父级并程分支程序中的子并程调用指令
- 与并程数据序（代表上下文数据）一起排序的子并程调用许可。

并程的第 N+1 个分支程序可以编程，控制和实现子并程的调用许可，并实现和并程数据序的混合。因此，N+1 并程是一种支持并程嵌套的结构。

显然，本发明的结构特征包括：

- 简单的处理 N+1 个并程分支程序的并程时分管理结构；
- 第 N+1 个并程分支程序建立并程数据序，实现运行同步和数据传送功能；
- 支持并程和并程间的嵌套关系；以及
- 并程 N+1 个分支程序是独立的模块化结构。

上述特征是序网计算机的多机并程优点移植到了单机的环境。

在单机[illegible]图 2A 表示图 1A 的并程模块结构和功能可以用单机的串型[illegible]现。为了实现和上下程序的连接，需要在图 2A 的基础上增加一个并程进入/退出管理模块 210，如图 2B 所示。

图 2A 实际上显示了并程已经具备一个微型化的进程（或线程）管理能力。因此，每一个实际应用的并程需要初始化。每一个并程有一个对应的并程标志区，初始化就是对全部并程标志作初始数值的填充处理，如在运行开始要建立 N+1 个并程分支程序的初始化结构参数，包括 N 个分支程序的数目、各分支程序的入口地址、以及其他初始化参量等。这些处理，需要一个并程进入管理模块。

并程的退出也需要根据情况做不同的处理。当并程作为一个子程序，运行完毕后退出，就和常规的子程序返回一样。如果并程是在全部分支程序悬挂的状态下退出，就可能有多种不同的原因。一种可能是该并程是由父级并程分支程序中的一个并程调用指令启动的，该并程全部分支程序悬挂实际上表示该父级并程分支程序进

入悬挂状态，父级并程分支程序需要退出，转其他的父级并程分支程序运行。另一种情况是本并程需要等待外部的输入信息，因此，就需要并程进入循环等待状态。第三种情况是并程在设计或运行中出了错，需要转入有控制的自动纠错处理或显示出错的处理。这些处理，需要一个并程退出的管理模块。

因此，根据本发明的实施例包括一个并程进入/退出管理模块 210。该并程进入/退出模块 210 和时分管理模块 201 在实际设计中可以组合在一起。

单机中，一个实用的并程封装结构包括模块 210 和模块 200。在用户编写 N+1 个并程分支程序后，并程的编译工具在编译过程中自动的为每一个并程添加并程进入/退出模块 210、时分管理模块 201、其中包括并程标志和全部并程数据标志的存储器空间，最后还要用三个子程序来替代用户编程中采用的共享存储器读写的表达格式。

和先有技术中，并程在软件中以图 1B 和图 1C 的方式出现，本发明的图 2B 的方式是一种更一般性的结构。

由于并程具备组合分支程序的能力、一般性结构，和可嵌套特征，本发明认为并程将成为今后建模或程序设计的一种通用元件。从子程序计算到系统程序，从操作系统到用户程序，到处都有或者是性能简化的、或者是性能加强的并程模块。

每一个并程都有一个并程数据区和一个并程专用标志区。并程数据区保存并程内部互连的并程数据，并且可以和并程外部进行交换。并程专用标志包括并程标志（图 2C）和并程数据标志（图 2D）。

图 2C 是并程标志结构图。并程标志支持软件实现时分处理过程。一个并程只有一组并程标志，包括并程类型标志 220，并程分支程序开关标志 221（N+1 bits），并程分支程序就绪/悬挂标志 222（N+1 bits），以及 N+1 组并程分支程序的悬挂状态保护 223（指令地址，寄存器数据，堆栈区内容等）。

并程类型标志 220 是一种表示并程的特征的标记。例如可以用于指示一旦全部并程分支程序发生悬挂，是出错原因，或是父级并程分支程序的悬挂原因，或是程序循环等待输入的原因等。标志位可以根据并程类型发展的需要来建立。

并程分支程序开关标志 221 表示并程分支程序进入运行和退出的状态，有 N+1 位，在并程初始进入时，N+1 个位全部打开 ON，一旦一个并程分支程序结束，则关闭该并程分支程序 OFF，全部并程分支程序关闭表示并程运行结束。

并程分支程序就绪/悬挂标志 222 表示并程分支程序的运行条件，共 N+1 位表示 N+1 个并程分支程序。一旦分支程序因为并程数据无效而不能运行，就进入悬挂状态。一旦并程数据有效，则该分支程序状态就修改为就绪。如果出现有分支程序开关是 ON，而全部分支程序状态为悬挂，表示并程没有结束，但无法运行。

每一个并程还有 N+1 组并程分支程序的悬挂状态保护 223，包括分支程序中断的地址，各寄存器的数据，堆栈区内容，以及其他返回需要的并程分支程序信息。

并程标志区是并程专属的，动态的，在并程启动前并程标志要加载，各并程分支程序的入口地址也要加载，并根据并程的分支程序结构数目进行打开（ON）设置。在悬挂和返回时需要保护和加载。但是当并程结束后，这些标志就没有用了。

图 2D 是一个并程数据标志结构图。并程数据标志用于软件实现并程分支程序的同步和数据传送过程。每一个并程数据（或并程数据块）都有一个对应的并程数据标志，包括一致化有效（1 bit），读有效（1 bit），曾读无效（N bits），和曾一致化无效（1 bit），四种标志。

一致化有效标志 225 共一位，表示该数据已经由某个并程分支程序写过，使该数据允许进行一致化操作。它在写数据后建立有效标记，并在一致化操作前测试，测试无效使并程一致化分支程序进入悬挂状态。一致化有效标志 225 也可以设置 N 位，指示出写该数据的分支程序。

读有效标志 226 共一位，表示该数据已经一致化，是有效数据，允许各并程分支程序读取。它在一致化操作后建立有效标记，并在读数据前测试，测试无效使该读无[illegible]分支程序进入悬挂状态。

曾读无效标志 227 共 N 位，表示该数据曾经被并程分支程序读过，但是数据无效。N 位分别表示 N 个并程分支程序的曾读无效。它们在读过程检测无效后建立标志，在写数据后测试，测试有效改悬挂的并程分支程序为就绪状态。

曾一致化无效标志 228 共一位，表示该数据曾经被执行一致化操作，但是当时数据无效，一致化实际没有执行。它在一致化检测无效后建立标志，在一致化操作成功后测试标志，若有标志，则悬挂状态的并程一致化分支程序标志为就绪状态。

并程数据标志在并程运行过程中需要，并程结束就不再有用了。因此，需要有事先的标志清理，事后的工作存储区释放。

对并程标志来讲，图 1B 所示的方法是不需要这些标志的，可以节省存储器空间和程序运行时间，结构也简单。通过引入并程标志和管理模块，用软件方法将并程功能全部移植进单机环境，使图 2A 和图 1B 的 N 个程序的建模几乎完全一致，但分支程序却保持了独立结构。同时，由于多了第 N+1 个程序的资源，本发明的并程将支持一些现在单机软件技术不能解决功能。

图 3A 是并程管理程序流程图。本流程图包括并程进入和退出管理模块 210 和时分管理模块 200。其功能是：

- 使并程对外呈一个串型程序段，可以用 Jump 或 Call 方式进入，以及用 return 方式退出。
- 实现并程 N+1 个分支程序的时分管理，支持同步。
- 实现并程标志的管理。
- 并程[illegible]的优先模式。
- 流程图处理下列并程管理标志：
- 并程标志管理（并程分支程序开关，悬挂/就绪，类型标志）。
- 并程分支程序现场保护存储区（全部并程分支程序的中断地址，寄存器，堆栈区内容等信息）。

流程图的工作原理如下：流程图有二个入口，分别为初始化处理入口 300，和并程分支程序悬挂或退出返回入口 302。初始化处理入口 300 由外部程序的调度来启动，并程分支程序悬挂或退出返回 302 来自分支程序中读数据无效或一致化数据无效的子程序，“分支程序退出返回” 302 来自分支程序运行结束后的返回指令。其中：

- 初始化处理300包括并程参量的加载处理和并程标志区的清0处理。初始化加载包括打开并程的N+1分支程序标志ON，设置各并程分支程序的入口地址，设置相应的寄存器数据初始值等；清并程标志区是将并程标志区和并程数据标志清0区。
- 模块302的入口表示并程运行需要时分管理模块201定义一个新的可执行并程分支程序。

二个入口经过301后，进入并程分支程序结束检查步骤303，检查并程的开关标志221。如果有一个并程分支程序是开ON，进入步骤305，进行分支程序的悬挂状态检查。如果全部为关OFF，表示并程运行已经结束，转并程结束步骤304，处理并程结束的记录以及和外部程序的连接方式。

并程分支程序的悬挂检查步骤305检查并程就绪/悬挂标志222。如果步骤305检查结果是全部并程分支程序处于悬挂状态，表示尽管有并程没有执行完，但该并程已不能运行下去了。转步骤306查找并程不能运行的原因，并根据原因处理转入步骤310，处理并程的退出。并程不能运行的原因可能是：

- 在正常运行情况下，表示该并程是父级并程分支程序中发动的并程调用，是并程嵌套。现在并程不能运行表示该父级并程分支程序的悬挂状态。因此，需要退回到父级并程的时分管理模块去处理，调度另外一个父级并程分支程序。因此，该并程要在携带悬挂标志的情况下退出，以便让该并程模块所在的父级并程分支程序进入悬挂。
- 在并程模块有错误的情况下，则进入出错处理。
- 程序设计为扫描循环，等待外部如键盘输入。
- 其他本发明不作进一步探讨的原因。

如果步骤305检查结果是有一个以上的并程分支程序处于就绪状态，则进入分支程序排队处理步骤307。分支程序排队处理步骤307选择一个就绪的并程分支程序，然后转选定分支程序的加载步骤308。选择方法实际上是一种处于就绪状态的分支程序的排队规则，可以设计各种优先规则。优先规则在编程时指定，在编译时实现。其中最简单的优先规则是循环排队的方法。步骤307按照循环顺序，进行并程分支程序就绪状态的检查。

步骤308是在步骤307确定了下一个运行的并程分支程序后，处理该并程分支程序参量的加载，恢复该并程分支程序先前悬挂时的状态，加载完成后，转入步骤309进入该选定的并程分支程序运行。

N+1并程分支程序的时分运行是并程的基本方法。根据不同的应用场合，可以对并程的结构作不同的修改和发展。时分处[illegible]既可以属于操作系统，也可以进入用户程序。

根据本图所表示的流程，一个熟练的软件工程师编制一个实际运行的程序是没有困难的，同时，在本图核心流程的基础上进行改动或修改也是可能的，可以想象的。变化包括各个方向，例如，一种时分方式的发展是在上述自动时分方法处理分支程序时，再加入一个时间片的切换条件。使得当一个并程分支程序（即一个进程）的运行时间不到时间片数值出现悬挂时，则自动切换；当一个并程分支程序（即一个进程）的运行时间超过时间片数值时，则强制切换。

根据先有技术序网的工作原理状况，要求在读数据指令，写数据指令和一致化指令过程中对数据进行有效状态的检测。在先有技术的多机环境下，检测结果是和一个中断程序相连接的。在本发明

中，软件模拟的子程序根据检测结果直接和相应的处理程序连接，实现并程分支程序的数据一致化处理和程序的同步处理。

本发明的图 3B，C，D 是一种采用软件来模拟三种序网指令及其处理的实施例。软件模拟数据令牌要求对每一个数据建立并程数据标志（如图 2D 所示）；并在并程数据的读、写和一致化操作时，增加检测标志的操作和根据标志位的处理操作。其结果软件模拟的读数据，写数据和数据一致化指令是三个独立的子程序。各子程序的流程图在图 3B、图 3C、图 3D 中展示，其中，并程分支程序的悬挂处理模块 202 也在子程序中用软件模拟了。

图 3B 是写数据子程序的流程图。并程的写数据指令在数值计算中是取一个并程数据的操作，在单机中是一个子程序，或是 C 语言环境下的一条语句。在用软件仿真写数据指令的情况下，子程序需要增加数据状态标志的处理，和一个 [illegible]无效”的判断，以及判断后的处理。

写数据子程序的流程图工作原理如下：写数据子程序首先进入写数据步骤 320，实现一个并程数据的写操作。然后进入“曾一致化无效”的检查步骤 321。步骤 321 程序检查该数据“曾一致化无效”标志 228。如果步骤 321 中该标志 228 为有效，表示该数据曾有过数据一致化操作，但因为一致化标志无效，使该一致化操作实际上没有能执行，导致了一致化并程分支程序进入悬挂状态。现在，由于写数据步骤 320 已经执行，数据已经有效了，可以进入步骤 323。步骤 323，处理一致化程序的就绪。所谓就绪，就是改就绪标志 222 中该一致化并程分支程序的状态位为就绪（原来是悬挂），然后转入步骤 322。如果步骤 321 中该标志 228 为无效，表示该数据没有执行过数据一致化操作，直接转步骤 322。步骤 322 是建立“一致化有效”标志 225，表示数据有效，使允许一致化并程分支程序 100

对该数据执行一致化的操作。然后，转步骤 324。步骤 324 是写数据子程序的返回。

图 3C 是数据一致化子程序的流程图。并程的数据一致化指令在数值计算中是一个并程数据的传送操作，也是确立一个并程数据有效的操作，在单机中是一个子程序。在用软件仿真写数据指令的情况下，子程序需要增加数据状态标志的处理，一致化令牌有效检查，和曾读无效检查，以及检查后的处理。

数据一致化子程序的流程图工作原理如下：数据一致化子程序首先进入步骤 330。检查该数据的“一致化有效”标志 225。如果步骤 330 中该数据的“一致化有效”标志无效，表示数据还没有被写过，因此，该并程分支程序不能运行，要悬挂。程序进入步骤 332。步骤 332 设置“曾一致化无效”标志 228，然后进入步骤 333 进行当前一致化分支程序[illegible]理。内容是保存一致化并程分支程序的当前运行状态，以便[illegible]后该并程分支程序返回时使用。步骤 333 是软件实现模块 202 的方法。然后转步骤 334 退出该并程分支程序，转并程分支程序悬挂入口 302，以便重新选择新的并程分支程序。

如果对该数据的“一致化有效”标志 225 检查后，该数据的“一致化有效”标志有效，表示数据已经被写过，数据的一致化可以运行。转步骤 331。步骤 331 执行数据和令牌的一致化操作。在单机软件模拟中，没有实际的数据传送，仅仅是“读有效”标志 226 的设置。转步骤 335 检查该数据的“曾读无效”标志 227， 曾读无效标志 227 有 N 位，分别代表 N 个不同的并程分支程序（第 N+1 个一致化并程分支程序用“曾一致化无效”标志）。如果步骤 335 中“曾读无效”标志 227 有效（指标志 227 的 N 个标志位中至少有一位有效，最多为 N 个标志位全有效），说明某些并程分支程序出现过曾读无效情况，并已经转入了悬挂状态。由于已经有步骤 331 执行一

致化操作，因此，转步骤 336，步骤 336 是根据“曾读无效”标志 227 的内容，使相关的并程分支程序状态从悬挂转入就绪，然后转步骤 337。如果步骤 335 中“曾读无效”标志 227 没有任何标志（指标志 227 的 N 个标志位中全部无效），则说明没有出现过曾读无效情况。此时便直接转步骤 337。

步骤 337 是数据一致化子程序的返回。

图 3D 是读数据子程序的流程图。并程的读数据指令在数值计算中是取一个并程数据的操作，在单机中是一个子程序。在用软件仿真读数据指令的情况下，子程序需要增加数据状态标志的处理，和一个数据有效性的判断，以及判断后的处理。

读数据子程序的流程图工作原理如下：读数据子程序首先进入步骤 340，检查该数据的“读有效”标志。如果步骤 340 中该数据的“读有效”标志有效，表示数据已经被写过，被一致化传送过，程序进入读数据步骤 341。步骤 341 是常规的单机程序的读数据方式，读数据后进入步骤 342。步骤 342 是读数据子程序的返回。如果步骤 340 中该数据的“读有效”标志无效，表示该数据现在无效（该并程分支程序要悬挂）。程序进入步骤 343。

步骤 343 是设置“曾读无效”标志 227，“曾读无效”标志 227 有 N 位，分别对应不同的并程分支程序。本步骤在该并程数据标志 227 内设置和该并程分支程序相应的“曾读无效”标志。然后转步骤 344。

步骤 344 是当前并程分支程序的悬挂处理。内容是保存并程分支程序的当前运行状态，以便在今后该并程分支程序返回时，可以从悬挂点切入继续运行。步骤 344 是软件实现模块 202 的方法。然

后转步骤 345。步骤 345 是退出该并程分支程序，转并程分支程序悬挂入口 302，重新选择新的并程分支程序。

图 3E 是并程的分支程序结构。并程有 N+1 个并程分支程序，PPc 350、PP1 351、PP2 352、PP3 354，它们都是独立的程序模块结构。独立性不仅表现在编程阶段，一直到编译以后的机器码阶段，上述 N+1 个并程仍旧有独立程序模块结构。各并程分支程序启动运行是由模块 201 指定。各并程分支程序运行结束，转入步骤 354。步骤 354 是一个转移到退出返回入口 302 的操作，表示一个并程分支程序运行结束，转交时分管理模块 201 处理。

N+1 个并程内部包含了三种特有的并程子程序，处理并程数据的读、写和一致化操作。由于在单机中全部用软件来模拟，每一个并程数据需要有相关的并程数据标志。一旦读或一致化子程序经测试，判定该并程分支程序不能继续执行，则由模块 202 处理并程分支程序的悬挂，然后退回到时分管理程序 201。一旦写或一致化子程序经测试，判定原来在该并程数据进入悬挂的其他并程分支程序条件已经满足，则子程序改其他处理并程分支程序的状态为就绪，然后返回。这些子程序处理的流程，在图 3B、图 3C、图 3D 中已经描述。一旦组成了一个 N+1 并程，并程分支程序就包含了这些子程序的调用特征。

本发明的软件模拟并程需要相关的并程标志。和图 1B 相比，它们是要在单机环境下保持并程结构和功能产生的增量。

N+1 个并程分支程序 207 是用户的编程界面。其中 N 个并程分支程序 PP1 351，PP2 352，PP3 353 是常规的单机编程，仅仅是加入共享存储器方式的并程数据读写。它们在编译时，改为用上述子程序格式来代替共享存储器读写。由于这是编程中的关键差别，因此，可以认为本发明的编程和习惯编程方式一致。

第 N+1 个并程分支程序 PPc 350 也是一个独立的程序。它和常规的程序不同，没有任何数值计算的工作。它包括一个并程数据的序，和将并程数据进行“一致化”的操作。由于如图 3C 所示，存在一个一致化操作的子程序 P_CONSIS_SUB，PPc 并程分支程序实际是对并程所包含的 m 个数据进行如下的操作:

```
START 123          ; 并程 123 的第 N+1 个分支程序进入
PP123_DATA1        ; 定义并程 123 的第 1 个并程数据
P_CONSIS_SUB       ; 并程 123 的第 1 个并程数据一致化
PP123_DATA2        ; 定义并程 123 的第 2 个并程数据
P_CONSIS_SUB       ; 并程 123 的第 2 个并程数据一致化
.....
PP123_DATAm        ; 定义并程 123 的第 m 个并程数据
P_CONSIS_SUB       ; 并程 123 的第 m 个并程数据一致化
END   123          ; 并程 123 的第 N+1 个分支程序结束
```

其中，PP123_DATA1 定义了一个编号为 123 的并程的第一个并程数据。实际上是定义了并程数据的相关参量供一致化子程序 P_CONSIS_SUB 的执行。

上述是一个最简单的，也是最基本的第 N+1 个程序的例子。其最大特点是它是一个程序，能驱动并程数据一致化的有序执行，起到了协调 N+1 个并程分支程序的同步运行。在结构上，并程数据和并程数据的序能独立出来。作为程序，并程数据一致化的有序执行

只是很简单的功能，还可以有许多可以发展的资源，例如实现并程数据的序的变化，增加或减少并程数据，以及插入子并程调用许可指令等等。

一个熟练的软件工程师编制程序实现上述N+1个并程分支程序的软件应该没有困难，同时，在本发明所示的流程图基础上进行改动或修改也是可能的。

先有技术和本发明并程结构的描述集中在并行流程图的计算功能上，但是本发明的真正所得的并非仅仅是一个并行流程图的数值计算内涵，而是并程在单机中的新结构所内含的价值。这种价值可能现在还没有一个类似于并行流程图的功能可以表达。它除了产生并程数据序、支持并程嵌套结构的特征外，还包括本发明的并程结构所产生的新资源。这是指：在用户继续面对 N（或 N+1）个独立模块结构的并程分支[illegible]下，出现一个一致化并程分支程序 PPc 350 的程序资源，[illegible]有富裕能力的并程管理模块（包括模块 201 和 210）资源。

本发明提出一些新资源的实施例如下：

1. 并程模块的“并程数据+序”特征

第 N+1 个并程分支程序执行并程数据序，使并程数据序成为并程封装的外部特征。也就是说，如果并程作为一个面向对象的类或对象的内部结构的话，它的封装的外部特征不只有数据一种要素，而转为具备并程数据和并程数据序二种要素。其结果是面向对象技术中的类或对象出现实时特征，表现为有时间来调制的事件的序。例如，这些数据的时间关系指示某个有序的系列化操作，是不能改变先后关系的。

"并程数据+序"能更好地反映对象的动态执行特征，有助于支持面向对象技术的创新。而现有面向对象的技术，类的封装仅有数据一种要素，相当于本发明的并程结构放弃了序要素的一个特例。

2. 并程分支程序的在线增减

在并程有了并程数据序特征情况下，一个符合并程数据序新并程分支程序可以方便的加入并程，相当于并程结构的一个旁路程序。旁路程序和并程有共享的并程数据序、有独立的结构、可以用于类的继承，数据采集，软件测试，运行监控等各种用途。

3. 在用户程序中，并程可以提高运行响应速度。

用户程序中有三段程序分别接受三个外来的数据，数据到达[illegible]的。在串型程序下，将在第一个数据位置停顿，而不能运行其他二个接收程序的准备段。如果改串型程序为本发明的有三个分支程序的并程结构，则程序将完成全部三个程序的接收准备阶段后，才进入等待。并且，一旦有一个数据先到，先到数据所在的并程分支程序就可以进入运行。

4. 并程是计算过程的一般性结构

本发明的并程结构第一次实现了并行程序的组装模块。其中，各分支程序的悬挂、就绪等类进程的处理方法进入了并程的内部结构。第 N+1 个分支程序突出了整个并程的并程数据序特征，实际上，第 N+1 个分支程序运行的并程数据序和并程的数值计算结果没有关系，其意义已经不在完成一个并行流程图的计算，而是表现并程计算过程的特征。本发明的并程是计算过程的一般性结构，通过结构参量的变化可以形成如图 1B，图 1C 这些不同的并程实施例。

一条单机调用指令就可以启用这个并程，使并程的应用极其方便。

5. 单个并程数据转为数据块变化。

并程数据结构转为数据块结构变化的并程结构差别在于：从图3D 所示的一个并程数据有一个并程数据标志结构转为一批并程数据设立一个并程数据标志的结构。可以通过设计一组新的子程序类型来实现，但在编程概念上仍可以是共享存储器的结构。

6. 其他

象多个并程间的嵌套结构、并行程序的并发调用等，都是本发明并程结构特有的功能，可以支持计算机软件的各种需求。

并程和并程调用产生了新的结构资源，但是在使用方法上和先有技术相似，一个熟练的软件工程师应该能理解和实现这些新资源的运用。并且随着研究的深入，更多的应用将会被开发和实现。

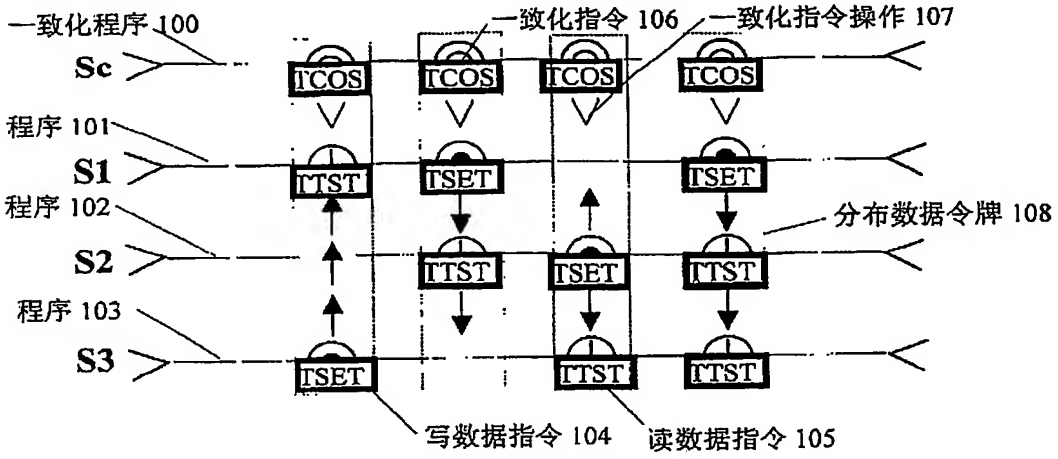
一致化程序 100
一致化指令 106
一致化指令操作 107
Sc
TCOS
程序 101
S1
TTST
TSET
程序 102
S2
分布数据令牌 108
程序 103
S3
写数据指令 104
读数据指令 105

图 1A

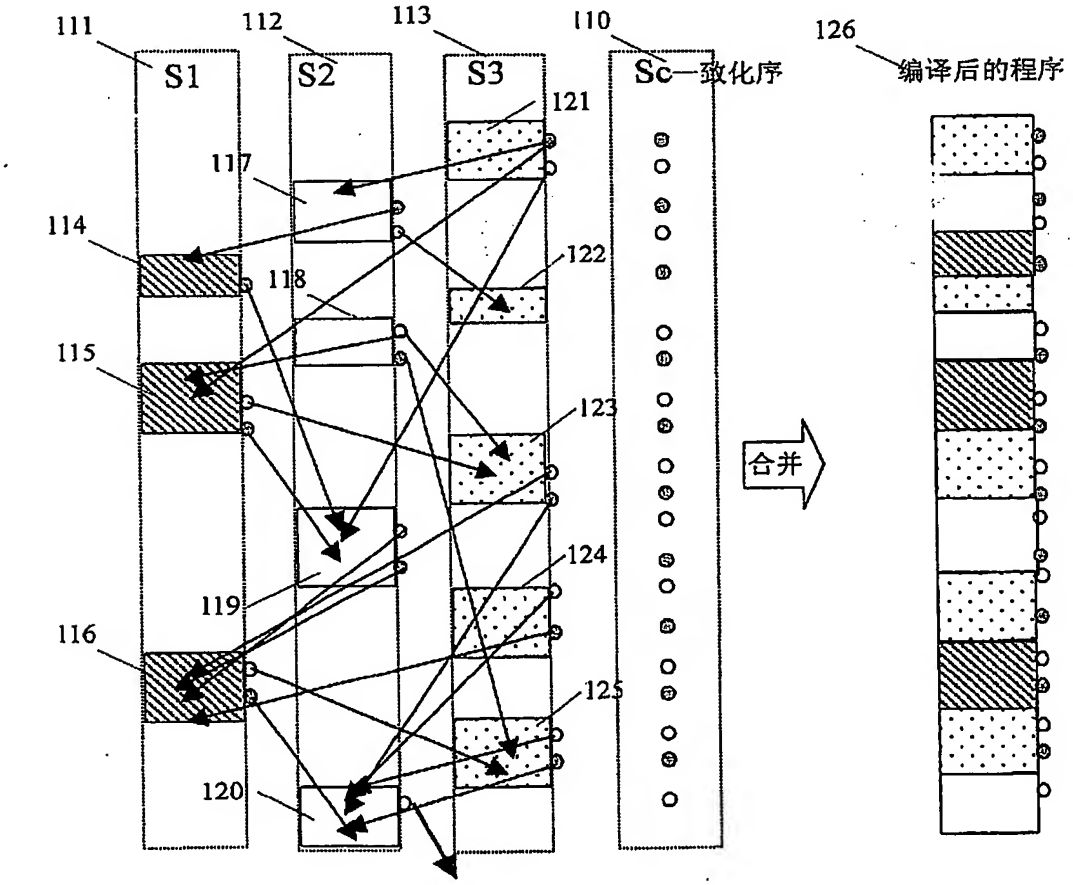
111
S1
112
S2
113
S3
110
Sc—致化序
126
编译后的程序
121
117
114
122
118
115
123
124
119
116
125
120
合并
数据传送
读
写
数据

图 1B

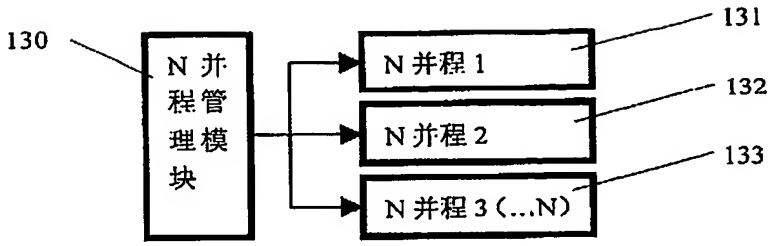
130
N 并程管理模块
N 并程 1
N 并程 2
N 并程 3（...N）
131
132
133

图 1C

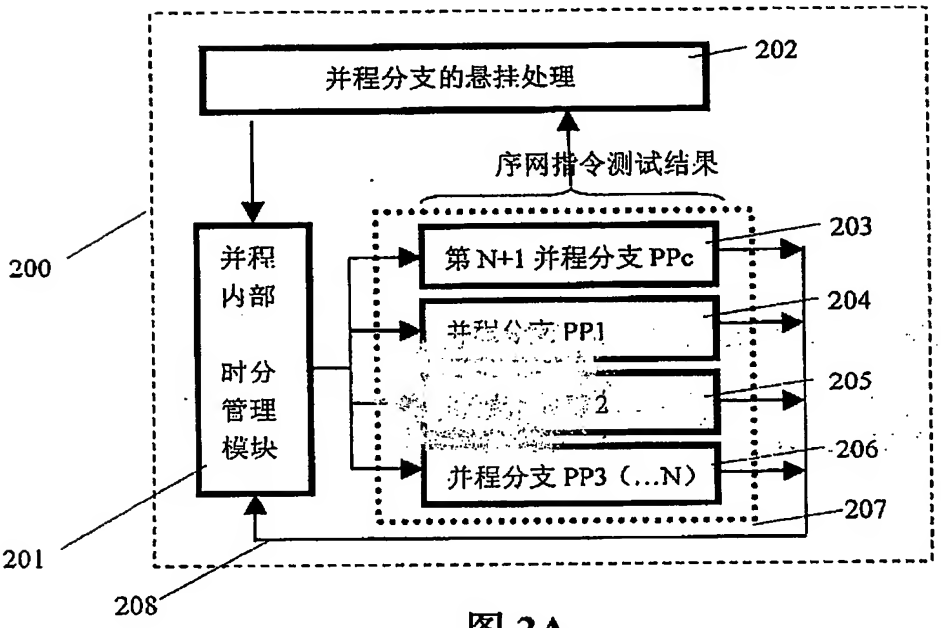
并程分支的悬挂处理
202
序网指令测试结果
200
并程内部
时分管理模块
第 N+1 并程分支 PPc
并程分支 PP1
并程分支 PP3（...N）
203
204
205
206
207
201
208

图 2A

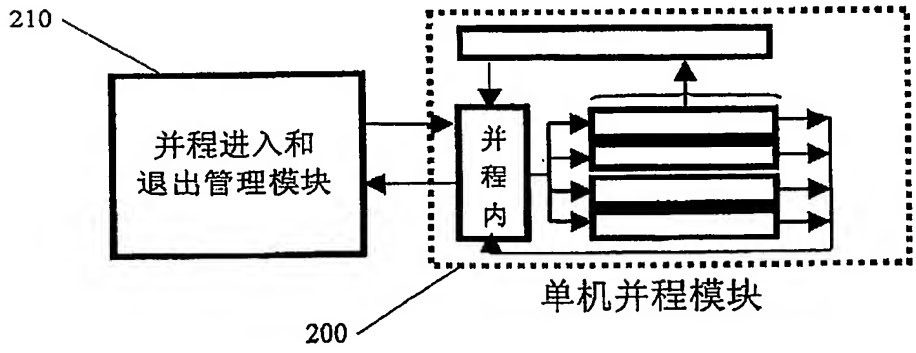
210
并程进入和退出管理模块
并程内
单机并程模块
200

图 2B

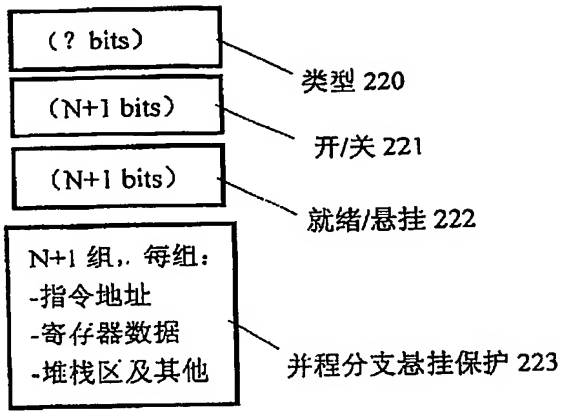
（？ bits）
类型 220
（N+1 bits）
开/关 221
（N+1 bits）
就绪/悬挂 222
N+1 组，每组：
-指令地址
-寄存器数据
-堆栈区及其他
并程分支悬挂保护 223

图 2C

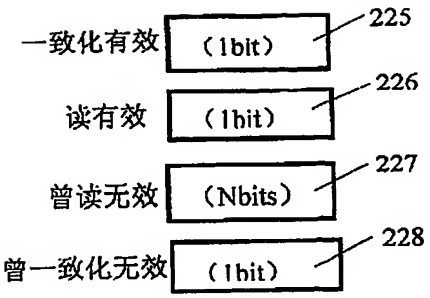
一致化有效
（1bit）
225
读有效
（1bit）
226
曾读无效
（Nbits）
227
曾一致化无效
（1bit）
228

图 2D

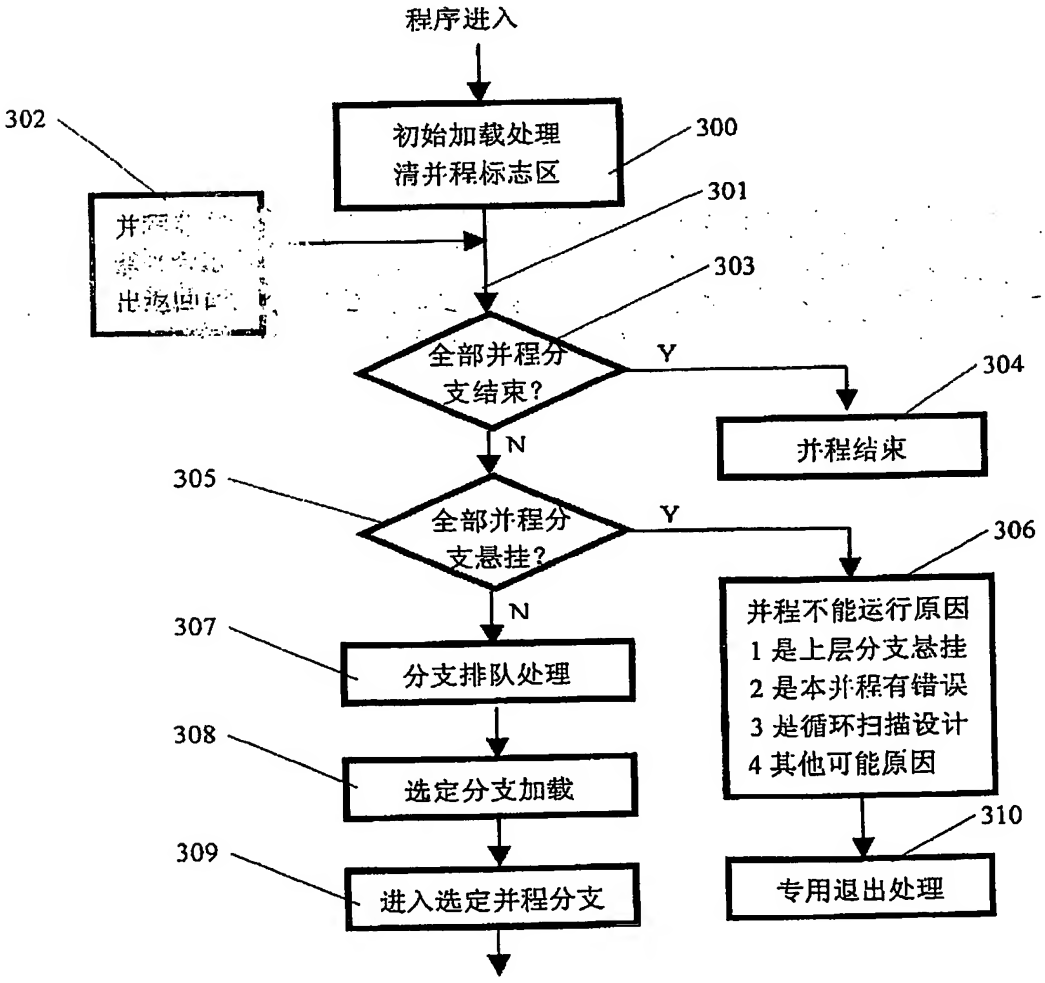
程序进入
初始加载处理
清并程标志区
300
302
301
303
全部并程分支结束？
Y
304
并程结束
N
305
全部并程分支悬挂？
Y
306
并程不能运行原因
1 是上层分支悬挂
2 是本并程有错误
3 是循环扫描设计
4 其他可能原因
N
307
分支排队处理
308
选定分支加载
309
进入选定并程分支
310
专用退出处理

图 3A

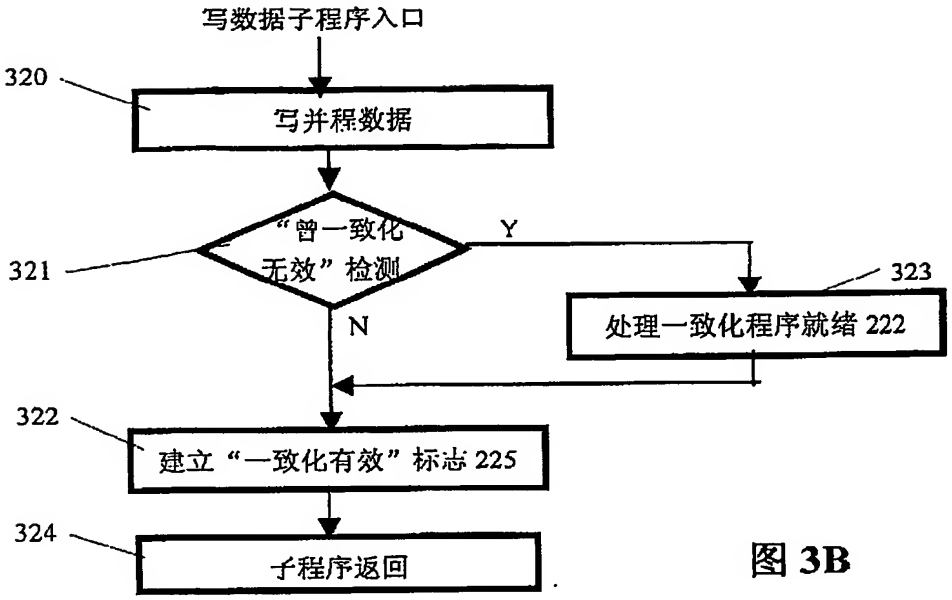
写数据子程序入口
320
写并程数据
"曾一致化无效"检测
321
Y
323
处理一致化程序就绪 222
N
322
建立"一致化有效"标志 225
324
子程序返回

图 3B

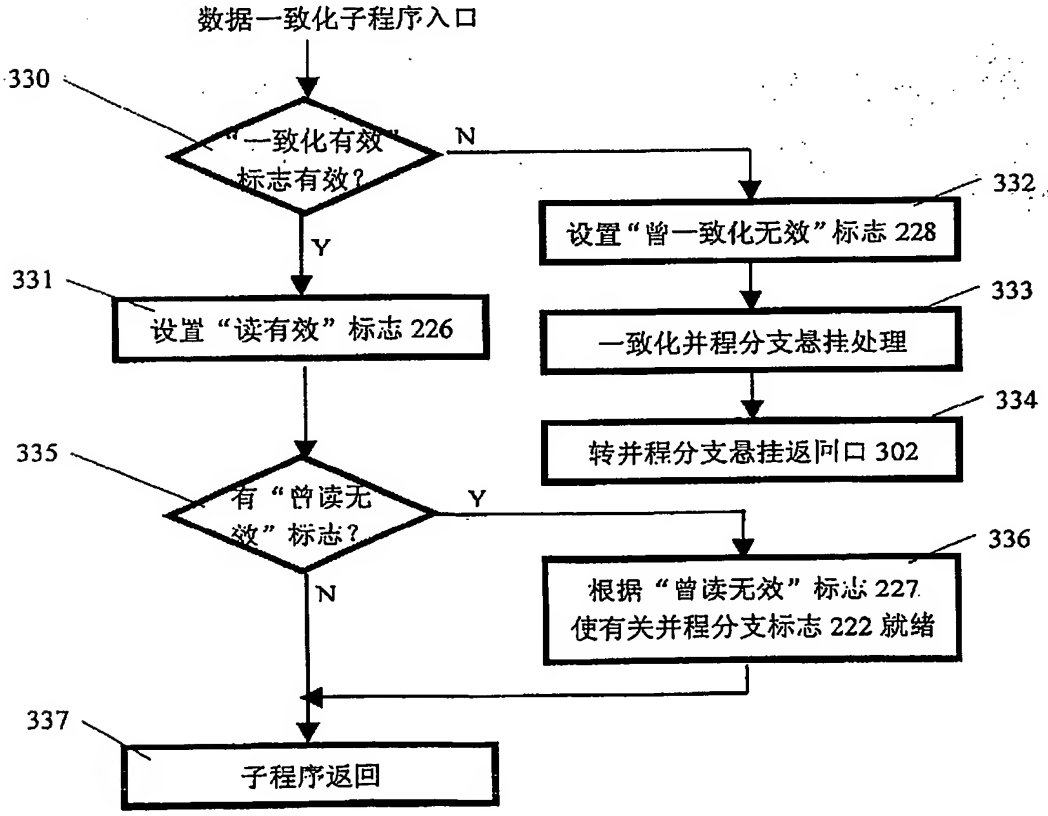
数据一致化子程序入口
330
"一致化有效"标志有效?
N
332
设置"曾一致化无效"标志 228
Y
331
设置"读有效"标志 226
333
一致化并程分支悬挂处理
334
转并程分支悬挂返回口 302
335
有"曾读无效"标志?
Y
336
根据"曾读无效"标志 227 使有关并程分支标志 222 就绪
N
337
子程序返回

图 3C

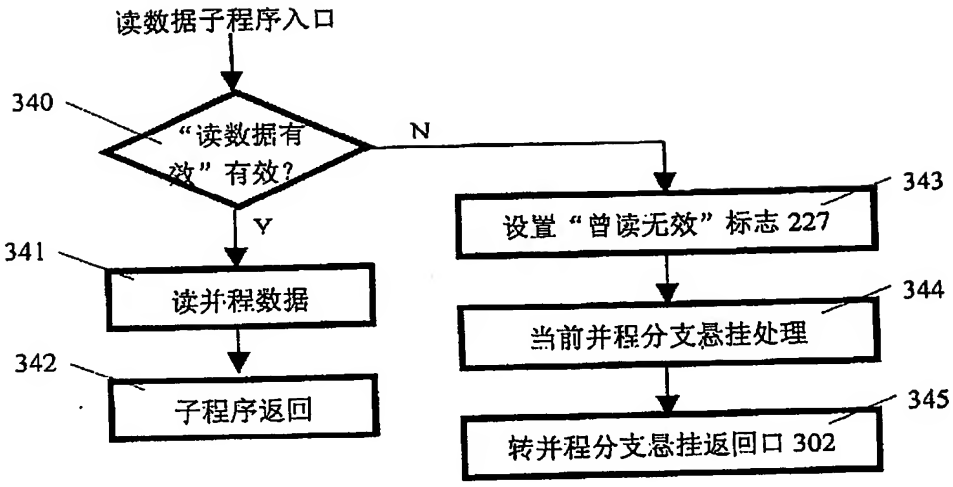
读数据子程序入口
340
“读数据有效”有效?
N
Y
341
读并程数据
342
子程序返回
343
设置“曾读无效”标志 227
344
当前并程分支悬挂处理
345
转并程分支悬挂返回口 302

图 3D

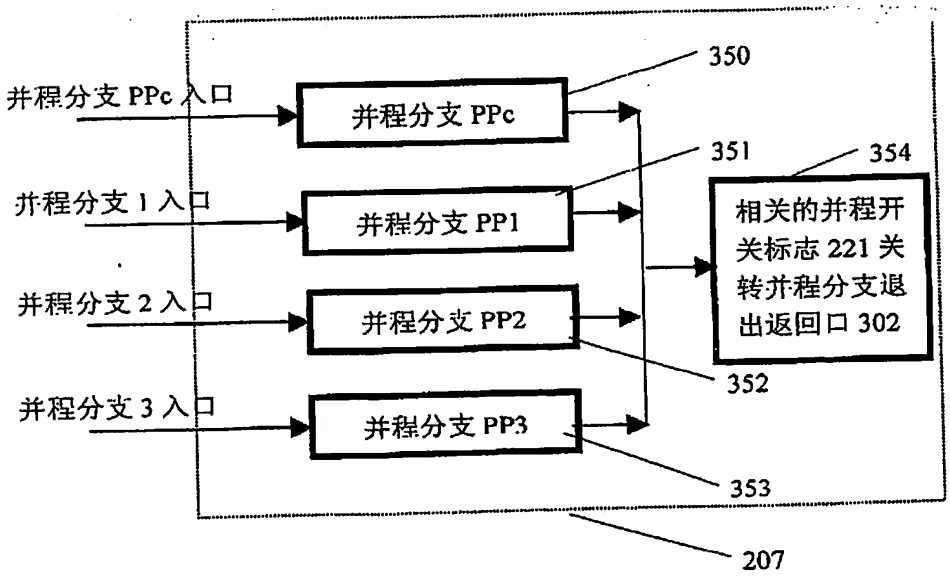
并程分支 PPc 入口
并程分支 PPc
350
并程分支 1 入口
并程分支 PP1
351
并程分支 2 入口
并程分支 PP2
352
并程分支 3 入口
并程分支 PP3
353
354
相关的并程开关标志 221 关转并程分支退出返回口 302
207

图 3E